An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software

Jim Smith

April 2004

TECHNICAL REPORT
CMU/SEI-2004-TR-013
ESC-TR-2004-013

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software

CMU/SEI-2004-TR-013
ESC-TR-2004-013

Jim Smith

April 2004

Integration of Software-Intensive Systems Initiative

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Acknowledgements

I’d like to thank Ceci Albert, Lisa Brownsword, and Wolfhart Goethert for starting me on this journey with their insight that readiness comprises several distinct aspects. I’m also grateful for invaluable comments received from Tricia Oberndorf, Ceci Albert, SuZ Garcia, Bob Ferguson, and Duane Hybertson. Lastly, I’d like to thank Claire Dixon for helping me to put this into some semblance of order.

Abstract

Defense acquisition policies require that program managers conduct technology readiness assessments for all critical technologies. Technology Readiness Levels (TRLs) are frequently used in performing these assessments. While there is considerable evidence to support the utility of using TRLs in assessing program risk, there are some difficulties in using TRLs with software. This report explores these problems as they apply to non-developmental items (NDI), including commercial off-the-shelf, government off-the-shelf, and open source software technology and products. The problems take four principal forms:

1. TRLs "blur" several aspects of technology and product readiness into a single number.
2. TRLs do not account for the criticality of a product or technology to the system as a whole.
3. TRLs don't account for software technology and product aging.
4. TRLs do not provide any means to deal with how the relative contributions of the various aspects of readiness vary throughout the life cycle of a system.

This report examines these issues in detail and proposes an alternative approach for determining product readiness of NDI software technology.

1 Background

1.1 Introduction

This report grew out of an earlier effort to develop a methodology for assessing the readiness of commercial software technologies and products, in the context of a massive system reengineering and modernization program being undertaken by an agency of the U.S. Department of Defense (DoD). Initial attempts, based on Technology Readiness Levels (TRLs), led to a realization that they provided insufficient insight into the readiness of software products and technologies—especially non-developmental items (NDI), including commercial off-the-shelf (COTS), government off-the-shelf (GOTS), and open source software (OSS) technology and products. As this realization grew, some of my colleagues developed and refined the notion that readiness comprises several distinct aspects, each of which can be measured and reasoned about independently. With that foundation, this report explores an alternative set of readiness criteria, as well as an assessment methodology that is better suited than TRLs for determining the readiness of NDI software product and technology for use in a system under development.

The remainder of this section will provide a brief overview of TRLs, followed by an explanation of the relationship between quality and readiness; the concept of readiness in a given context is also introduced. The next section will explore some of the issues which surface when applying the TRL methodology to software products: COTS, GOTS, and other NDI (including OSS). The rest of the report will describe an alternative approach for determining the readiness of COTS/GOTS/NDI software (hereinafter referred to as "NDI" software) and define a methodology for performing these assessments and analyzing the results. An example is provided to illustrate these ideas, and some conclusions and suggestions for future research are presented.

1.2 TRLs: A Brief Description

Technology Readiness Levels (TRLs) were first used by the National Aeronautics and Space Administration (NASA) Goddard Space Flight Center in the late 1980s, as part of an overall risk assessment process; the TRLs, and their definitions, are provided in Appendix A [Eisman 97]. By the early 1990s, TRLs were routinely used within NASA to support technology maturity assessments and consistent comparisons of maturity between different technologies.

The TRL methodology was incorporated into NASA Management Instruction (NMI) 7100 as an integral part of the technology planning process [Mankins 95].

The DoD adopted TRLs for use in risk assessments in 1999, and the Air Force Research Lab (AFRL) has adapted the NASA TRLs for use in assessing the readiness of critical technologies for incorporation into weapon systems [GAO 99]. Current DoD guidance requires the use of TRLs (or an equivalent methodology) as part of an overall system risk assessment [DoD 03]. The TRLs range from 1-9, with 9 signifying the highest degree of readiness. The General Accounting Office (GAO) recommends that technologies be matured to at least TRL 6 prior to initiating an acquisition program; attainment of TRL 7 is recommended prior to starting the System Development and Demonstration (SD&D) phase [GAO 99]. TRL 6 is defined as

> *System/subsystem model or prototype demonstration in a relevant environment*

and represents "…a major step in the level of fidelity of the technology demonstration" that goes "…well beyond ad hoc, 'patch cord' or discrete component level breadboarding." [Mankins 95]. Similarly, TRL 7 is defined as

> *System prototype demonstration in an operational environment*

This represents a significant step beyond TRL 6, requiring an actual prototype system in the intended operational environment (e.g., tactical aircraft or ground vehicle). Interestingly, the GAO notes that most leading commercial firms (other than space systems technology firms) required new technologies to be matured to TRL 8 ("Actual system completed and qualified through test and demonstration") prior to using them in new product developments [GAO 99]. As numerous GAO reports have documented, DoD acquisition programs frequently enter into SD&D with critical technologies at TRLs as low as 2 or 3, which often results in significant cost growth and schedule delays [GAO 99, 00, 03a, 03b].

In response to the increasingly important role that software plays in DoD systems, and recognizing the "hardware-centric" focus of NASA's TRLs, both the Army Communications Electronics Command (CECOM) and AFRL have recently been working on extending the definitions of TRLs to encompass software. As an example, CECOM's draft TRL 6 for software expands on the NASA definition (shown above), to read

> *Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in software demonstrated readiness. Examples include testing a prototype in a live/virtual experiment or in a simulated operational environment. Algorithms run on processor of the operational environment are integrated with actual external entities. Software releases are "Beta" versions and configuration controlled. software sup-*

> *port structure is in development. VV&A* [verification, validation and accreditation] *is in process* [Graettinger 02].

The draft CECOM software TRLs are provided in Appendix B. In addition, an employee at AFRL has developed a tool to automate the process of determining hardware and software TRLs through an interactive questionnaire [Graettinger 02]. This "customization" of TRLs is not unique to the DoD: there are also several examples of extending TRLs into such widely varying domains as information assurance technology, "practice-based technologies," biomedical technology, modeling and simulation, and conflict resolution, as well as one example of defining a "new" TRL (TRL 10) [Graettinger 02, Graettinger 03, DOE 00, Hetrick 02, McCleskey 01].

1.3 Relationship Between Quality and Readiness

Understanding the need for an alternative to TRLs first requires an understanding of what is meant by "readiness." Readiness, as used in this report, is a measure of the suitability of a software technology or product for use within a larger software-intensive system *in a particular context* (e.g., development of a management information system or sustainment of a deployed tactical information processing system). In other words, the readiness of the software product or technology reflects some measure of the risks of using it in the larger system: higher readiness denotes lower risk; lower readiness, higher risk. This can best be illustrated through the use of a recognized quality model, such as ISO/IEC 9126-1 (Software engineering—Product quality—Part 1: Quality model) [ISO 2001]. In this model, software quality is defined in terms of six external and internal quality characteristics, each with a number of sub-characteristics (Figure 1), and four "quality in use" characteristics (Figure 2). Readiness, then, can be thought of as representing some non-linear combination of these characteristics and sub-characteristics, in the context of a particular system.

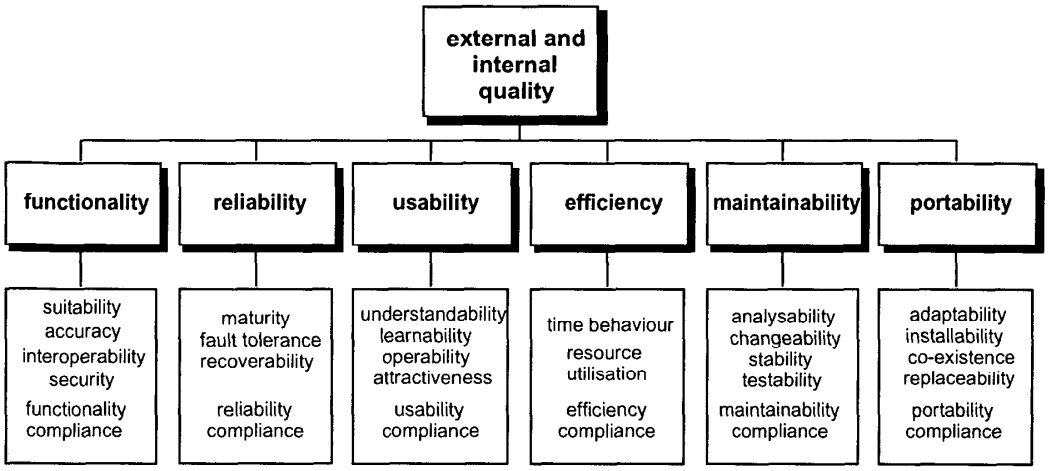

Figure 1: Quality Model for External and Internal Quality.

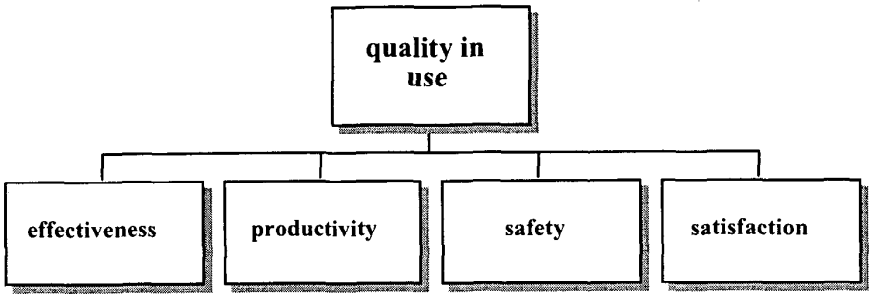

Figure 2: Quality Model for Quality in Use

It is important to note that "readiness" and "maturity"—though frequently used interchangeably—are *not* the same thing. A mature product may possess greater or lesser readiness for use in a particular system context than one of lower maturity. Numerous factors must be considered, including the relevance of the products' operational environments (e.g., usage patterns, timeliness/throughput requirements, etc.) to the system at hand, product/ system architectural mismatch, as well as other factors that will be discussed later in this report.

1.4 Understanding Readiness in Context

To better understand how context influences the determination of readiness in a software product or technology, a picture may be useful. In their paper, Hanakawa and colleagues model the knowledge growth experienced by an organization during software development. The resulting knowledge growth can be represented by a sigmoid (s-shaped) curve, several examples of which are shown in Figure 3 [Hanakawa 98]. We can extend this model to a software-intensive system acquisition or development, and equate "knowledge" with some

measure of system maturity, such as requirements satisfaction or technical performance measure (TPM) improvement. We then find that a typical acquisition or development will mature slowly during initial concept exploration and technology development until some critical point is reached (e.g., fundamental science is understood or algorithms validated) at which point progress becomes more rapid. As a system moves towards greater maturity, and most—though probably not all—requirements are satisfied, progress tapers off. In Hanakawa's model, the exact shape of this curve is dependent on the

- statistical distribution of tasks (e.g., requirements to be satisfied or program milestones)
- degree of task difficulty
- knowledge/competence of the organization to perform the tasks
- rate at which knowledge is accumulated through task performance

Thus, every acquisition and development will result in a unique "maturity profile."

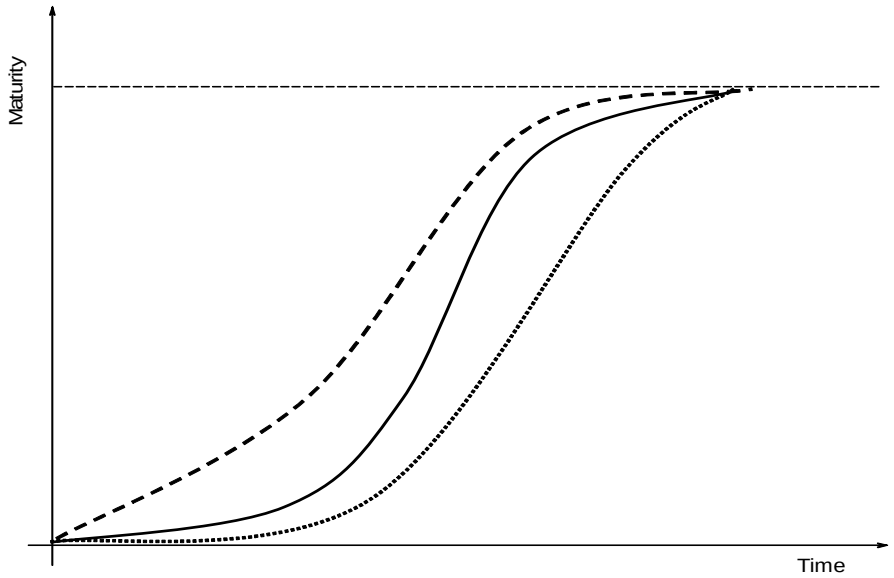

Figure 3: Characteristic "Maturity Growth" Curves

In Figure 4, representative DoD acquisition phases and milestones are overlaid on this maturity growth curve. Within each of these phases, the slope and curvature (concave or convex) of the sigmoid curve reflects the changing rate of maturity growth within the program.

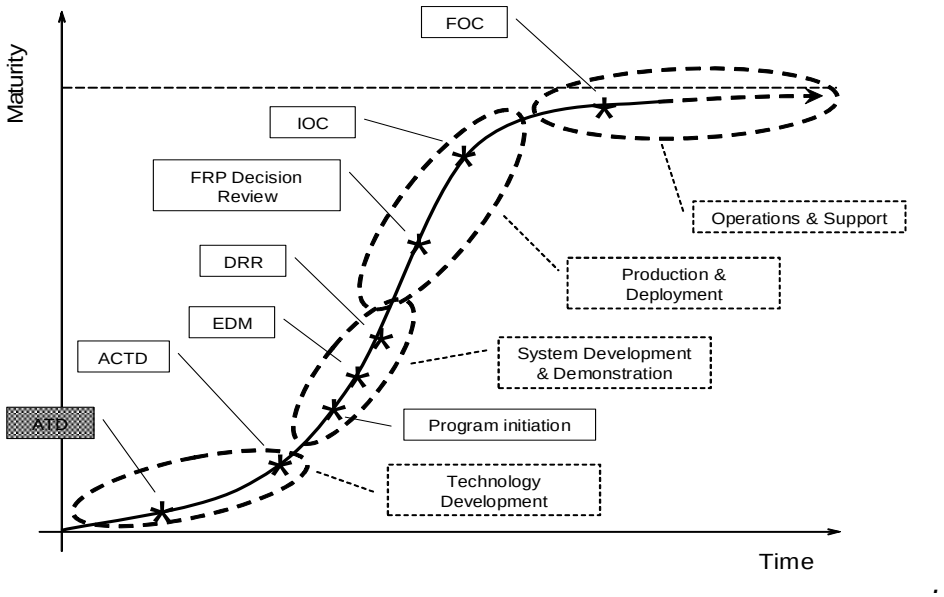

Figure 4: Representative DoD Acquisition/Development Program Phases and Milestones Overlaid on a Typical Maturity Growth Curve

These maturity growth curves provide some insight into the system development context, and permit an understanding of how the individual contributors to product or technology readiness vary in importance during the course of the program. For example, early in a program's life cycle, the fact that a software technology or product is projected to be unsupported sometime during the system's operational lifetime is probably of much less significance than if that product or technology is so closely tied to the system's architecture or implementation that replacing it would send you "back to the drawing board." On the other hand, during the post-deployment sustainment phase, the impending retirement of a product or technology may become as important—or possibly more important—than how closely tied it is to the system's design. The key to this approach is that, while the absolute values of the individual contributors to product or technology readiness cannot be defined, it is possible (in fact, it is necessary) to articulate the importance of one aspect (e.g., importance) relative to another, using "fuzzy" definitions like "as important as," or "much less important than." This provides the basis for the evaluation framework described later in this report.

2 The "Problem" With TRLs

Given the origin of TRLs, it is unsurprising that organizations experience difficulty in using them to assess the readiness of software-based technologies and products. Characteristics of TRLs include the blurring, or blending together of multiple components of readiness; the lack of any built-in mechanism to deal with issues such as the "criticality" of a technology or product; NDI product "aging"; and varying sensitivities to different contributors to readiness experienced at different points in the development/acquisition life cycle. These characteristics complicate TRL use in assessing the readiness of software technology, especially for NDI software products. These issues are discussed in more detail in the following sections.

2.1 "Blurring" Different Contributors to Readiness

One of the difficulties with using TRLs in programmatic and technical risk assessments is the manner in which TRL definitions combine several different aspects of, or contributors to, technology and product readiness. For example, CECOM's draft software TRLs defines TRL 7 as follows:

> *Represents a major step up from TRL 6, requiring demonstration of an actual system prototype in an operational environment… Algorithms run on processor of the operational system and are integrated with actual external entities. Software support structure is in place. Software releases are in distinct versions. Frequency and severity of software deficiency reports do not significantly degrade functionality or performance. VV&A completed* [Graettinger 02].

Thus, TRL 7 combines aspects from across all the product external quality characteristics: for example, functionality ("Algorithms run on [the] processor of the operational system and are integrated with actual external entities."), maintainability ("Software support structure is in place."), and reliability ("Frequency and severity of software deficiency reports do not significantly degrade functionality or performance"), as well as several quality-in-use characteristics. The manner in which these combine makes it difficult, if not impossible, to understand how any one aspect contributes to, or influences the overall readiness of the product or technology.

2.2 Product/Technology Criticality

Just as importantly, TRLs leave out such considerations as the degree to which the technology is critical to the overall success of the system (including how difficult it would be to replace it, or assume some fall-back posture, should the technology in question prove unacceptable), or the suitability of the technology in question to its intended use within the system.

Some programs have attempted to deal with this effect through the use of correction factors to adjust the TRL for a given technology for, say, the "criticality" of that technology to the success of the system (as measured by the percentage of the total system capability provided by the technology in question), or the technical complexity of the technology. For example, a program may adjust a TRL downward by some amount if a particular technology or product comprises more than some threshold, measured as a percentage of the functionality of the system [Wong 00]. Other techniques include normalizing technology readiness to the relevant environment for the different life-cycle phases of an acquisition or development (e.g., for a laboratory "bench top" test, a product or technology with a TRL of 3 or 4 may be acceptable) [GAO 99, Graettinger 02, Wong 00].

2.3 NDI Software Aging

TRLs were designed to measure the maturation of technologies as a way to gauge their readiness for use in a specified context. In this view, a technology (e.g., as used in a spectrometer) that has been "flight proven" through successful operation in space would be evaluated as being at TRL 9. Absent any changes to the way in which the technology is employed, it remains at TRL 9.

Software, on the other hand, is continually changing. As Vic Basili notes, a COTS software product generally "…undergoes a new release every eight to nine months, with active vendor support for only its latest three releases" [Basili 01]. Furthermore, software ages as a result of maintenance activities. In their paper, Stephen Eick and colleagues discuss three mechanisms of maintenance-induced software aging:

1. "Span of changes," which is shown to increase over time
2. "Breakdown of modularity," which manifests loss of architectural integrity of the software
3. "Fault potential," which indicates the probability that modifications introduce new faults into the software [Eick 01]

Compounding these effects is the fact that a system developer using NDI software as part of a larger system has little or no control over the scope or timing of these changes. Similarly, other forms of NDI software (i.e., GOTS, OSS) experience analogous decay processes.

The bottom line is that software—especially NDI software products—begins to decay as soon as it is released. TRLs provide a way to measure the increasing maturation, but lack any way to deal with this continual degradation in software readiness.

2.4 Readiness In Context

The above-mentioned issues, coupled with the realization that context varies throughout the life cycle of a system, introduce a fourth problem area: different aspects of technology or product readiness contribute, in varying degrees, to system risk at different times, and for different types of acquisitions. For example, the fact that there is an “end of life” announcement for a product that is critical to a given system is probably more significant if the system is fielded and operational, than if the system is a laboratory prototype not intended for operational use.

3 An Alternative Approach

The previous section outlined some of the issues related to using TRLs in assessing the readiness of NDI software products and technologies. The remainder of this report will introduce a new approach that addresses these issues, and show how this can complement and extend the current TRL process to provide greater insight into the technical and programmatic risks facing a program.

3.1 Readiness Attributes

Given that the readiness of a software product or technology reflects some combination of quality characteristics in a specific context, then reasoning about readiness requires the definition of some attributes of readiness. Addressing the issues raised in the previous discussions on TRLs, these attributes should

- Provide coverage of the quality attributes most important to determining readiness.
- Be "orthogonal." In other words, one criterion should not be a function of another one.

While TRLs combine various quality aspects in a way that it is impossible to directly discern the contributions of any particular aspect to the overall readiness of a product or technology, they *do* provide useful insights into two key contributors to readiness:

1. degree of functionality provided
2. fidelity of the environment (to the intended operational environment) in which this functionality has been demonstrated

Other key contributors to readiness that are missing from the TRLs include product/technology criticality in the context of the system under consideration, and the effects of software aging. There are two aspects of aging that are of particular interest in this context: the maturity of a product or technology—which varies by its "domain" (e.g., COTS, GOTS, or other NDI, like OSS)—and its availability.

The remainder of this section will describe a set of proposed readiness attributes, with definitions for various "levels" within each attribute, and attempt to show how these attributes satisfy (or at least improve upon TRLs) the requirement for coverage of the salient quality char-

acteristics. Orthogonality of these attributes, and a proposed evaluation framework, will be discussed in the next section.

3.1.1 Requirements (Functional and Non-Functional) Attribute

This attribute describes how well the requirements, including functional (e.g., throughput, accuracy, latency) as well as non-functional (e.g., reliability, maintainability) allocated to a given software product or technology are satisfied by it. For functional requirements, this includes not only how many requirements are satisfied, but also any provided functionality that is *not* required. As previously mentioned, this is one of the two attributes which is derived from the definitions of TRLs. Table 1 provides the definition and a brief description of the Requirements attribute.

Table 1: Requirements Attribute Definitions

Requirements (R)	
Evaluation	***Definition***
Ideal (I)	"Perfect" fit between requirements and product/technology capabilities. In other words, the product does exactly what is required: nothing more, nothing less. This rarely occurs in practice.
Good (G)	Requirements satisfied, but there are some minor "fit" issues. These may include some capabilities in the product that are not required, and may represent a potential vulnerability, or could result in unanticipated (or possibly undesired) usage patterns.
Fair (F)	Deficiencies in one or more second/third-tier requirements, with workaround possible. In DoD acquisition parlance, this might be the case where all threshold requirements and key performance parameters (KPPs) are satisfied while some objective requirements are not, but there are operational and/or technical workarounds for the missing functionality.
Limitations (L)	Deficiencies in one or more second/third tier requirements, with no workarounds. This is similar to "Fair" case, except that there are no acceptable workarounds for the missing functionality.

Table 1: Requirements Attribute Definitions (cont')

Major Limitations (M)	One or more major requirements unsatisfied; system performance degraded. This is the case where one or more threshold requirements or KPPs are unsatisfied, but there are workarounds which provide the required functionality, albeit with some degradation.
Unsatisfactory (U)	One or more major requirements unsatisfied with no workarounds. This is the case where some threshold requirements and/or KPPs are not satisfied, and there are no acceptable operational or technical alternatives. In this case, the software product or technology under consideration is unsuited for the intended use.

There are a number of techniques to determine the "fit" between the allocated requirements and the capabilities of a product or technology, including the "Risk Misfit" process described by Wallnau and colleagues and the "Gap Analysis" methodology described by Ncube and Dean [Wallnau 02, Ncube 02].

3.1.2 Environmental Fidelity Attribute

This attribute describes how faithfully the environment in which the software product under evaluation has been demonstrated reproduces the target operational environment. This provides some insight into a product's ability to satisfy the allocated requirements based on observed performance in another context. Table 2 provides definitions and brief descriptions for environmental fidelity.

Table 2: Environment Fidelity Attribute Definitions

Environment (E)	
Evaluation	***Definition***
Full (F)	Subject product/technology demonstrated through use in the actual operational environment under "fully stressed" conditions (e.g., maximum required transaction rates, throughput, network limitations, heat, humidity)

Table 2: Environment Fidelity Attribute Definitions (cont')

Partial (P)	Use in a less than fully stressed operational environment demonstrated. This may represent a relaxation from the maximum required throughput, number of simultaneous users, and so forth, expected in the intended operational environment.
Simulation (Si)	Use in a simulated operational environment demonstrated. This is analogous to using a SPECmark® benchmark to characterize processor performance. For any valid conclusions to be drawn, the simulated environment must reflect the most important aspects of the actual operational environment.
Comparable (C)	Product/technology demonstrated through actual use in a comparable environment. For example, in evaluating a word processor, a commercial office environment may represent a sufficiently close approximation for a military office environment to reasonably infer the word processor's performance in that environment. On the other hand, if the word processor is going to be used in an operational command-and-control setting, then any judgments about its projected performance in a tactical environment, based on its performance in a commercial office environment, are meaningless.
Integration (I)	Software product integrated with other components in a development/integration environment. While functionality can be demonstrated, no attempt is made to simulate the relevant characteristics (e.g., number of simultaneous users, throughout) of the target operational environment.
Standalone (St)	Product used in a standalone environment. Other components/ subsystems/ systems are represented by low-fidelity simulations, or are simply "stubbed" (i.e., represented by non-functional—or "stub"—software code) to permit standalone operation. No meaningful conclusions can be drawn from operation of a software product in isolation—other than it does, in fact, run.

® SPECmark is a registered trademark of the Standard Performance Evaluation Corporation.

3.1.3 Product/Technology Criticality Attribute

This attribute is concerned with the degree to which the target system is dependent upon, or inseparable from the product or technology. For example, if the system is architected and partitioned so that the only interface between a product under evaluation and the target system is a simple asynchronous messaging interface, then the criticality of the product to the system is probably minimal. On the other hand, if the system depends on some proprietary capabilities contained within the product for its correct performance, or the interface consists of numerous, complex application programming interfaces (APIs), then the ability of the system developer to substitute another product is diminished—and the criticality of the product to the system is correspondingly greater. Table 3 contains representative levels, with brief explanations, for the criticality attribute.

Table 3: Product/Technology Critical Attribute Definitions

Criticality (C)	
Evaluation	***Definition***
Minimal (Mi)	At least one alternate product/technology can be easily substituted within the target system.
Low (L)	At least one alternate can be substituted; reintegration required with minimal software changes.
Moderate (Mo)	At least one alternate can be substituted; moderate reintegration required with pervasive software changes necessary.
Strong (S)	Substitution possible; significant architectural and/or implementation changes required, limited to a single aspect or partition of the system.
High (H)	Significant, wide-ranging architectural and/or implementation changes required; good candidate for re-factoring/re-design.
Fixed (F)	No flexibility: any changes to the product/technology under evaluation would require a complete redesign of the system.

As with requirements, there are several techniques that can be employed to assess the impact, or criticality, of a product or technology upon a system. Some of these include Options Analysis for Reuse (OAR), Mining Architectures for Product Lines (MAP), and Quality Attribute Workshop (QAW) [Bergey 01, Bergey 03, O'Brien 02].

3.1.4 Product Aging: Availability and Maturity Attributes

There are a couple of aspects to product "aging": First, there is the availability of the product. The Availability attribute provides some insight into this aspect by comparing a product's lifespan with the requirements of the system under development. Is it available now? When needed? For how long? If it is being retired, has a replacement been announced? Table 4 provides definitions and brief descriptions for the Availability attribute.

Table 4: Product Availability Attribute Definitions Table

Availability (A)	
Evaluation	***Definition***
Lifespan (L)	Product/technology available over the intended lifespan of the system under development. Note: this will almost never occur in practice.
Probably	Available by system "need date," but will probably be replaced during the system's life.
End-Of-Life (EOL) With Replacement (Ewr)	Available by system "need date," but product End-Of-Life (EOL) with replacement announced.
EOL Without Replacement (Ewor)	Available by system "need date," but EOL without replacement announced.
Alternate (A)	Not available by system "need date," but suitable alternate exists in the interim.
Unavailable (U)	Not available by system "need date," and no suitable alternate exists.

There are a number of actions that a system developer or acquirer can undertake to keep abreast of product availability. These include maintaining an active "market watch" capability,

either in-house, or augmented by external expertise. Other related activities include participating in relevant user groups and industry associations, as well as reading professional and market publications.

The second aspect of product aging is the maturity of the software product or technology. Unlike the case with the other attributes, there are several distinct modes, or domains, of NDI software with their own maturation mechanisms, each of which have differing implications to readiness.

While these three domains represent different maturation mechanisms, some rough equivalence across domains can be made. Table 5 provides definitions of these levels for each domain, with their corresponding equivalent maturity levels.

Table 5: Maturity Attribute Definitions

Maturity (M)			
Evaluation	***NDI Software Domain-Specific Definitions***		
	Commercial Off-The-Shelf (COTS)	***Government Off-The-Shelf (GOTS)***	***Open Source Software (OSS)***
Off-the-shelf (OTS)	Widespread commercial use; available as COTS	System has achieved Full Operational Capability (FOC), and is in sustainment.	Product in large-scale public use
Deployed (D)	Limited or first commercial use	System at Initial Operational Capability (IOC)	Product in limited public use
System Test (S)	Product undergoing public beta/release candidate testing	System in Operational Test and Evaluation (OT&E)	Product undergoing public beta/release candidate testing
Subsystem/Component Test (Su)	Product undergoing limited or private testing	System in Developmental Test and Evaluation (DT&E)	Product undergoing limited or private testing

Table 5: Maturity Attribute Definitions

Prototype (P)	"Engineering tool" or otherwise not intended as a consumer product; "opportunistic" re-use	System in development	Product under development
Concept (C)	Product exists only in marketing brochures (also known as "vapor-ware").	System planned/ budgeted, but development not yet started.	Product development announced, but not yet started

3.2 Evaluation Framework

We've seen that readiness is defined by multiple attributes and their importance relative to one another. Multi-Criteria Decision Making (MCDM) theory provides numerous methods for determining the optimal solution in the presence of multiple criteria. These methods fall into two broad classes: Multi-Objective Decision Making (MODM) and Multi-Attribute Decision Making (MADM), with subclasses based on the data used (i.e., deterministic, stochastic, or "fuzzy") and the number of decision makers (i.e., single or group) [Triantaphyllou 98]. MODM applies to problems in which the decision space is continuous; MADM, in contrast, is used in decision problems with discrete decision spaces. The methodology described in this report falls into the deterministic, single decision maker MADM class. Saaty's Analytic Hierarchy Process (AHP), supported by several commercially available tools (e.g., Expert Choice), is recommended as an evaluation approach [Saaty 80].

AHP defines a process for evaluating multiple criteria, using a hierarchical structure (i.e., goal, attributes and sub-attributes, and alternatives) and pair-wise comparisons to determine the alternative that best satisfies the desired goal. The use of ordinal values, such as "*x* is much more important than *y*" or "*x* has roughly the same importance as *y*," works well in the context of software-intensive system acquisition and development where cardinal values (e.g., "Criticality" = 7.5) cannot be defined with any degree of confidence. One issue with using AHP is that the evaluation criteria must be orthogonal for the results to be valid. In other words, one criterion cannot be dependent on another criterion. For example, when deciding what car to buy, a couple of possible evaluation criteria schemes are

Scheme 1 – Purchase cost, fuel economy, body color

Scheme 2 – Purchase cost, taxes, insurance cost

Ignoring the question of whether *either* of these schemes will lead you to an optimal choice, it is clear that the criteria in Scheme 1 are relatively orthogonal: cost, fuel economy and color are more-or-less unrelated to each another. Scheme 2, on the other hand, exhibits a positive correlation between the evaluation criteria. Thus, the criteria contained in Scheme 2 would be unsuited for evaluation by the Analytic Hierarchy Process. The criteria described in this report, on the other hand, do present at least the *appearance* of orthogonality: none of the attributes (i.e., criticality, requirements satisfaction, product availability, product maturity, and environmental fidelity) are expressed in terms of any other attribute, nor does the evaluation of an attribute imply anything about any other attribute.

While AHP provides a method to reason about the contributions of various attributes to satisfying a desired goal, neither AHP nor the approach described in this report define how the relative rankings of the criteria are obtained. Just as the SEI Capability Maturity Model® (CMM®) framework leaves the definition of appropriate processes to the implementing organization, this framework leaves the criteria evaluation definitions to the developing or acquiring organization.

3.3 An Example Application

To see how this could work in practice, a very simple hypothetical system provides the context within which a single software technology/product choice is examined. First, the evaluation is made using TRLs alone, then is repeated using the criteria and evaluation framework described in this report. Finally, the results of the two approaches are compared.

In this system, two NDI software products are being considered for potential use. The first of these, “Product A,” has the following characteristics:

- It is nearing deployment, and is currently undergoing release candidate testing.
- All of the system threshold requirements, and most of the objective requirements allocated to this component are satisfied. The missing functionality can be provided through the use of some operational workarounds.
- The product can be replaced fairly easily. That is, it is sufficiently decoupled from the rest of the system that changes in this product should not require changes elsewhere in the system.
- It is projected to be available by the “need date” for the system under development, but will probably have to be replaced sometime during the development system’s life cycle.
- Its capabilities have been demonstrated in an environment that partially replicates the intended operational environment for the final system.

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Similarly, “Product B” exhibits the following traits:

- It is currently available as a COTS product.
- All threshold requirements are satisfied. Most objective requirements are satisfied, but some functions are missing and cannot be satisfied through any combination of work-arounds.
- The integration of this product and the system would necessitate a moderate reintegration effort, with widespread—though relatively moderate—software changes to the target system if the product had to be replaced.
- The product is available now—and will be available when needed for the development system—but there has been an “end of life” announcement, with a replacement planned by its developer.
- Its capabilities have been demonstrated in the target system’s intended operational environment.

3.3.1 TRL Assessment

Applying CERDEC’s draft software TRLs to evaluate the readiness of these two products, using the TRL calculator from AFRL, results in the following:

Product A: Evaluated as being at TRL 7
Product B: Evaluated as being at TRL 9

3.3.2 Alternative Assessment

The first step in this process is to determine the relative importance of the criteria in context. The context for a given system development is determined by the interactions of many complex variables, and cannot be ascertained by the application of any “cookbook” or prescriptive process. Among the factors to be considered in this determination are

- Where is the system in its life cycle?
- What is the development program’s risk tolerance?
- How important is it for the NDI product to be stable?

For this example, the relative importance of the criteria was determined to be as follows:

- Criticality (C) is slightly more important than Requirements (R). In other words, it is somewhat less important in this context that the product satisfy every requirement allocated to it, and more important that the system not be too dependent on any particular product choice.

- R is significantly more important than either Maturity (M) or Environmental Fidelity (E). This means that, for this stage in the system's development, it is much less important that the product be a true "off-the-shelf" product, or that it has been demonstrated in the intended operational environment, than it is for it to satisfy the allocated requirements or be easily replaced.
- M and E are, in turn, more important than Availability (A). This means that the likelihood that the product will be replaced during the life of the system is less important than its level of "productization," or how closely its demonstration environment matches that of the target system.

These relations can be expressed as

$$C > R >> \{M, E\} > A$$

In the AHP, this relation is converted into a Pair-wise Comparison matrix (PCM), where each entry in the matrix represents the comparison between the row attribute (X) and each column attribute (Y) using the following values [Saaty 80]:

X Compared to Y	***X Preferred to/more important than Y***	***Y Preferred to/more important than X***
Equally preferred	1	1
Moderately preferred	3	1/3
Strongly preferred	5	1/5
Very strongly preferred	7	1/7
Extremely strongly preferred	9	1/9

(To express comparisons falling between these levels, the intermediate values 2, 4, 6, and 8—as well as 1/2, 1/4, 1/6, and 1/8—can be used.) So, to express the relation "C is somewhat more important than R" in the PCM, a "2" would be entered in the PCM element that corresponds to the comparison between the "C" and "R" attributes (first row, second column). In the same way, the complete PCM for the relative importance of the attributes can be expressed as

$$\begin{array}{c} \\ C \\ R \\ A \\ M \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} C & R & A & M & E \end{array} \\ \begin{bmatrix} 1 & 2 & 8 & 6 & 6 \\ 1/2 & 1 & 6 & 4 & 6 \\ 1/8 & 1/6 & 1 & 1/2 & 1/2 \\ 1/6 & 1/4 & 2 & 1 & 1 \\ 1/6 & 1/4 & 2 & 1 & 1 \end{bmatrix} \end{array}$$

Since the lower triangular elements are simply the inverse of the upper triangular elements in a PCM, (i.e., $a_{i,j} = 1/a_{j,i}$ for $i > j$), it is simpler to represent the PCM by its *strictly upper triangular matrix* elements:

$$\begin{array}{c} \\ C \\ R \\ A \\ M \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} C & R & A & M & E \end{array} \\ \begin{bmatrix} 1 & 2 & 8 & 6 & 6 \\ 1/2 & 1 & 6 & 4 & 6 \\ 1/8 & 1/6 & 1 & 1/2 & 1/2 \\ 1/6 & 1/4 & 2 & 1 & 1 \\ 1/6 & 1/4 & 2 & 1 & 1 \end{bmatrix} \end{array} \Rightarrow \begin{array}{c} \\ C \\ R \\ A \\ M \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} C & R & A & M & E \end{array} \\ \begin{bmatrix} & 2 & 8 & 6 & 6 \\ & & 6 & 5 & 6 \\ & & & 1/2 & 1/2 \\ & & & & 1 \\ & & & & \end{bmatrix} \end{array}$$

The second step is to determine, within each of these attributes (i.e., C, R, A, M, and E), the relative preference/importance of the various levels within the context of a particular development. In this example, comparing the importance/desirability between a rating of "Minimal" for the Criticality attribute, and other levels within that attribute, resulted in a determination that "minimal" was

- moderately preferable to "low"
- strongly preferred to "moderate"
- very strongly preferred to "strong"
- somewhat more strongly preferred to "high" than to "strong"
- extremely more preferable than "fixed"

Completing the remaining comparisons, these judgments are then converted to a PCM:

	Mi	L	Mo	S	H	F
Mi		**3**	5	**7**	8	9
L			3	**5**	7	9
Mo				3	5	7
S					3	5
H						3
F						

This process is repeated for the remaining attributes (i.e., R, A, M, and E).

The next step is to evaluate each candidate product against each readiness attribute, resulting in a separate PCM for the candidates for each attribute. As discussed in an earlier section, this methodology neither prescribes nor proscribes any particular evaluation techniques: each development program is unique, and the implementation of this approach must be tailored accordingly. In this example, evaluating both products against the Criticality attribute results in the PCM:

$PCM_{Criticality}$

	A	B
A	1	6
B	1/6	1

Similarly, the remaining product/attribute PCMs are calculated:

$PCM_{Requirements}$

	A	B
A	1	4
B	1/4	1

$PCM_{Availability}$

	A	B
A	1	2
B	1/2	1

$PCM_{Maturity}$

	A	B
A	1	1/6
B	6	1

$PCM_{Environment}$

	A	B
A	1	1/4
B	4	1

Finally, applying the AHP with these PCMs produces weighted scores for the candidate products as shown:

Product A: 0.654

Product B: 0.346

3.3.3 Comparison of Results

From this *extremely* simple example, the effect of system and development context on product readiness is apparent. Using the existing (draft) software TRL definitions and the AFRL TRL calculator, Product B—which is available as a COTS product, and has been used in the target system's intended operational environment—is determined to be at a higher degree of readiness than Product A (TRL 9 versus TRL 7). When context is taken into account—reflecting management and engineering estimations about the relative importance of the readiness attributes, as well as value judgments about preferences within each attribute—Product A is seen to have higher readiness.

4 Conclusions and Next Steps

While there is a growing body of evidence that using TRLs as part of an overall risk assessment can lead to an improved understanding of the technological and programmatic risks in a system development or acquisition, there are several difficulties in applying "traditional" TRLs to the evaluation of software technologies. This is especially true for NDI, including COTS, GOTS, and OSS, where TRLs neither provide any way to discriminate between mature technologies or products, nor take into account the inevitable decay which all software experiences. Finally, the existing TRL framework lacks any explicit mechanism to deal with various aspects of the system and development context, including the time-varying effects of the various contributors to technology and product readiness.

The methodology described in this report provides an alternative to using TRLs for NDI software products and technologies that directly addresses these shortcomings. The methodology allows the evaluation criteria to be tailored to the particulars of any system development, including judgments about acquisition and development risk. As a result, a more nuanced determination of product or technology readiness is possible. On the other hand, considerably more effort is required to perform this evaluation than simply assessing TRLs.

So far, this methodology has not been applied to an actual system development and, thus, remains purely theoretical. It is planned, over the next year or so, to apply this to one or more case studies to see how well this approach is able to "predict the past." After some refinement, it should then be possible to pilot this methodology in an actual system development. If Mary Shaw is correct in her assertion that "It takes a good 20 years from the time that work starts on a theory until it provides serious assistance to routine practice," then there is some time remaining [Shaw 90].

Appendix A: Technology Readiness Levels

Table 6: Technology Readiness Levels and Descriptions

[Graettinger 02]

Technology Readiness Level	Description
1. Basic principles observed and reported.	Lowest level of technology readiness. Scientific research begins to be translated into applied research and development. Examples might include paper studies of a technology's basic properties.
2. Technology concept and/or application formulated.	Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.
3. Analytical and experimental critical function and/or characteristic proof of concept.	Active research and development is initiated. This includes analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.
4. Component and/or breadboard validation in laboratory environment.	Basic technological components are integrated to establish that they will work together. This is relatively "low fidelity" compared to the eventual system. Examples include integration of "ad hoc" hardware in the laboratory.
5. Component and/or breadboard validation in relevant environment.	Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so it can be tested in a simulated environment. Examples include "high fidelity" laboratory integration of components.

Table 6: Technology Readiness Levels and Descriptions (cont.')

6. System/subsystem model or prototype demonstration in a relevant environment.	Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high-fidelity laboratory environment or in simulated operational environment.
7. System prototype demonstration in an operational environment.	Prototype near, or at, planned operational system. Represents a major step up from TRL 6, requiring demonstration of an actual system prototype in an operational environment such as an aircraft, vehicle, or space. Examples include testing the prototype in a testbed aircraft.
8. Actual system completed and qualified through test and demonstration.	Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation of the system in its intended weapon system to determine if it meets design specifications.
9. Actual system proven through successful mission operations.	Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation. Examples include using the system under operational mission conditions.

Appendix B: CERDEC Draft Software TRLs

Table 7: CERDEC Draft Software Technology Readiness Levels and Descriptions [Graettinger 02]

Technology Readiness Level Description	**Technology Readiness Level Description**
1. Basic principles observed and reported	**HW/S:** Lowest level of technology readiness. Scientific research begins to be translated into applied research and development. Examples might include paper studies of a technology's basic properties. **SW:** Lowest level of software readiness. Basic research begins to be translated into applied research and development. Examples might include a concept that can be implemented in software or analytic studies of an algorithm's basic properties.
2. Technology concept and/or application formulated	**HW/S/SW**: Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.
3. Analytical and experimental critical function and/or characteristic proof of concept	**HW/S**: Active research and development is initiated. This includes analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative. **SW:** Active research and development is initiated. This includes analytical studies to produce code that validates analytical predictions of separate software elements of the technology. Examples include software components that are not yet integrated or representative but satisfy an operational need. Algorithms run on a surrogate process or in a laboratory environment.

Table 7: CERDEC Draft Software Technology Readiness Levels and Descriptions (cont.')

4. Component and/or breadboard validation in laboratory environment	**HW/S**: Basic technological components are integrated to establish that they will work together. This is relatively "low fidelity" compared to the eventual system. Examples include integration of ad hoc hardware in the laboratory. **SW**: Basic software components are integrated to establish that they will work together. They are relatively primitive with regard to efficiency and reliability compared to the eventual system. System software architecture development is initiated to include interoperability, reliability, maintainability, extensibility, scalability, and security issues. Software integrated with simulated current/legacy elements as appropriate.
5. Component and/or breadboard validation in relevant environment	**HW/S**: Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so it can be tested in a simulated environment. Examples include "high fidelity" laboratory integration of components. **SW**: Reliability of software ensemble increases significantly. The basic software components are integrated with reasonably realistic supporting elements so that it can be tested in a simulated environment. Examples include "high fidelity" laboratory integration of software components. System software architecture established. Algorithms run on a processor(s) with characteristics expected in the operational environment. Software releases are "Alpha" versions and configuration control is initiated. Verification, Validation, and Accreditation (VV&A) initiated.
6. System/subsystem model or prototype demonstration in a relevant environment	**HW/S**: Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high-fidelity laboratory environment or in a simulated operational environment. **SW**: Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in software demonstrated readiness.

Table 7: CERDEC Draft Software Technology Readiness Levels and Descriptions (cont.')

	Examples include testing a prototype in a live/virtual experiment or in a simulated operational environment. Algorithms run on processor of the operational environment are integrated with actual external entities. Software releases are "Beta" versions and configuration controlled. Software support structure is in development. VV&A is in process.
7. System prototype demonstration in an operational environment	**HW/S**: Prototype near, or at planned operational system. Represents a major step up from TRL 6, requiring demonstration of an actual system prototype in an operational environment such as an aircraft, vehicle, or space. Examples include testing the prototype in a testbed aircraft. **SW**: Represents a major step up from TRL 6, requiring the demonstration of an actual system prototype in an operational environment, such as in a command post or air/ground vehicle. Algorithms run on processor of the operational environment are integrated with actual external entities. Software support structure is in place. Software releases are in distinct versions. Frequency and severity of software deficiency reports do not significantly degrade functionality or performance. VV&A completed.
8. Actual system completed and qualified through test and demonstration	**HW/S**: Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation of the system in its intended weapon system to determine fit meets design specifications. **SW**: Software has been demonstrated to work in its final form and under expected conditions. In most cases, this TRL represents the end of system development. Examples include test and evaluation of the software in its intended system to determine if it meets design specifications. Software releases are production versions and configuration controlled, in a secure environment. Software deficiencies are rapidly resolved through support infrastructure.

Table 7: CERDEC Draft Software Technology Readiness Levels and Descriptions (cont.')

9. Actual system proven through successful mission operations	**HW/S**: Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation. Examples include using the system under operational mission conditions. **SW**: Actual application of the software in its final form and under mission conditions, such as those encountered in operational test and evaluation. In almost all cases, this is the end of the last "bug fixing" aspects of the system development. Examples include using the system under operational mission conditions. Software releases are production versions and configuration controlled. Frequency and severity of software deficiencies are at a minimum.

References

[Basili 01] Basili, V. & Boehm, B. “COTS-Based Systems Top 10 List.” *IEEE Computer* (May 2001): 2-4. <http://www.cs.umd.edu/projects /SoftEng/ESEG/papers/82.80.pdf> (2001).

[Bergey 01] Bergey, J.; & Fisher, M. *Use of the Architecture Tradeoff Analysis Method (ATAM) in the Acquisition of Software-Intensive Systems* (CMU/SEI-2001-TR-009, ADA396096). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports /01tn009.html>

[Bergey 03] Bergey, J.; O’Brien, L.; & Smith, D. *Application of Options Analysis for Reengineering*[SM] *(OAR*[SM]*) in a Lead System Integrator (LSI) Environment* (CMU/SEI-2003-TN-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications /documents /03.reports/03tn009.html>.

[Bilbro 01] Bilbro, J. *Technology Readiness*. <http://netcssi.nasa.gov/SLI_files /JimBilbro_TechnologyReadinessAssessment.ppt> (2001).

[DoD 03] Department of Defense. *Operation of the Defense Acquisition System).* <http://hfetag.dtic.mil/docs/DoD_5000-Brief-5-20-2003.ppt> (2003).

[DOE 00] Department of Energy. *Modeling and Simulation Technologies Future Combat System Workshop*. <http://www.amso.army.mil /2004-topics/fcs/feb-conf/overview.ppt> (2000).

[Eick 01] Eick, S.; Graves, T; Karr, A; Marron, J.; & Mockus, A. *Does Code Decay? Assessing the Evidence from Change Management Data*. <http://www.cs.umd.edu/class/spring2003/cmsc838p/Evolution /decay.pdf> (2001).

[Eisman 97] Eisman, M & Gonzales, D. *Life Cycle Cost Assessments for Military Transatmospheric Vehicles*. Santa Monica, CA: Rand Corporation, 1997. <http://www.rand.org/publications/MR/MR893/> (1997).

[GAO 99] General Accounting Office. *Better Management of Technology Development Can Improve Weapon System Outcome*. <http://www.gao.gov/archive/1999/ns99162.pdf> (1999).

[GAO 00] General Accounting Office. *Joint Strike Fighter Acquisition—Development Schedule Should Be Changed to Reduce Risks.* <http://www.gao.gov/new.items/ns00132t.pdf> (2000).

[GAO 03a] General Accounting Office. *Challenges and Risks Associated with the Joint Tactical Radio System Program.* <http://www.gao.gov/new.items/d03879r.pdf> (2003).

[GAO 03b] General Accounting Office. *Space Acquisitions: Committing Prematurely to the Transformational Satellite Program Elevates Risks for Poor Cost, Schedule, and Performance Outcomes*. <http://www.gao.gov/new.items/d0471r.pdf> (2003).

[Graettinger 02] Graettinger, C.; Garcia, S.; Siviy, J.; Schenk, R.; & Syckle, P. *Using the Technology Readiness Levels Scale to Support Technology Management in the DoD's ATD/STO Environment* (CMU/SEI-2002-SR-027, ADA407785) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02sr027.html>.

[Graettinger 03] Graettinger, C.; Garcia, S.; & Ferguson, J. *TRL Corollaries for Practice-Based Technologies*. <http://www.sei.cmu.edu/products/events/acquisition/2003-presentations/graettinger.pdf> (2003).

[Hanakawa 98] Hanakawa, H.; Morisaki, S.; & Matsumoto, K. "A Learning Curve Based Simulation Model for Software Development." (350-359) *Proceedings of the 20th International Conference on Software Engineering*, Kyoto, Japan, April 19-25, 1998. Washington, DC: IEEE Computer Society, 1998. <http://portal.acm.org/citation.cfm?id=302198&dl=ACM&coll=portal> (1998).

[Hetrick 02] Hetrick, M. *Conflict Management Methodology Readiness Levels (CMRL), Adapted From NASA Technology Readiness Levels (TRLs).* <http://cmsupport.org/Publications /CMSTechnologyReadinessLevels.htm> (2002).

[ISO 01] International Organization for Standardization. *ISO/IEC 9126-1 Software Engineering—Product Quality—Part 1: Quality Model*. Geneva, Switzerland: June 15, 2001.

[Mankins 95] Mankins, J. *Technology Readiness Levels – A White Paper*. <http://advtech.jsc.nasa.gov/downloads/TRLs.pdf> (1995).

[McCleskey 01] McCleskey, C. *Vision Spaceport—Renewing America's Space Launch Infrastructure and Operation*. Kennedy Space Center, FL: NASA, 2001. Available through <http://www.cctcorp.com/techpapers.htm> (2001).

[Ncube 02] Ncube, C. & Dean, J. "The Limitations of Current Decision-Making Techniques," 176-187. *Proceedings of the First International Conference on COTS-Based Software Systems*. Orlando, FL, Feb. 4-6, 2002. New York, NY; Springer-Verlag, 2002.

[O'Brien 02] O'Brien, L. & Smith, D. *MAP and OAR Methods: Techniques for Developing Core Assets for Software Product Lines from Existing Assets* (CMU/SEI-2002-TN-007, ADA403805). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents /02.reports/02tn007.html>.

[Saaty 80] Saaty, T. *The Analytic Hierarchy Process*. New York, NY: McGraw-Hill, 1980.

[Seacord 03] Seacord, R.; Plakosh, D.; & Lewis, G. *Modernizing Legacy Systems—Software Technologies, Engineering Processes, and Business Practices*. Boston, MA: Addison-Wesley, 2003.

[Shaw 90] Shaw, M. "Prospects for an Engineering Discipline of Software." *IEEE Software* (November 1990): 15-24

[Triantaphyllou 98] Triantaphyllou, E.; Shu, B.; Sanchez, N.; & Ray, T. “Multi-Criteria Decision Making: An Operations Research Approach,” 175-186. *Encyclopedia of Electrical and Electronics Engineering*, New York, NY: John Wiley & Sons, 1998.

[Wallnau 02] Wallnau, K.; Hissam, S.; & Seacord, R. *Building Systems from Commercial Components*. Boston, MA: Addison Wesley, 2002.

[Wong 00] Wong, B. *NASA Cost Symposium – Multivariate Instrument Cost Model-TRL (MICM-TRL)*. Goddard Space Flight Center, FL: NASA, 2000. <http://ipao.larc.nasa.gov/symposium /MICM-TRL-Wong.pdf> (2000).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	April 2004	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
An Alternative To Technology Readiness Levels for Non-Developmental Item (NDI) Software	F19628-00-C-0003

6. AUTHOR(S)

Jim Smith

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2004-TR-013

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2004-013

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Defense acquisition policies require that program managers conduct technology readiness assessments for all critical technologies. Technology Readiness Levels (TRLs) are frequently used in performing these assessments. While there is considerable evidence to support the utility of using TRLs in assessing program risk, there are some difficulties in using TRLs with software. This report explores these problems as they apply to non-developmental item (NDI) software technology and products, including commercial off-the-shelf, government off-the-shelf, and open source software. The problems take four principal forms:

1. TRLs "blur" several aspects of technology and product readiness into a single number.
2. TRLs do not account for the criticality of a product or technology to the system as a whole.
3. TRLs don't account for software technology and product aging.
4. TRLs do not provide any means to deal with how the relative contributions of the various aspects of readiness vary throughout the life cycle of a system.

This report examines these issues in detail and proposes an alternative approach for determining product readiness of NDI software technology.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software, Technology Readiness Levels, TRL, Acquisition, Risk, commercial off-the-shelf, COTS, government off-the-shelf, GOTS, Non-Developmental Item, NDI, Open Source Software, OSS	51

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL